Stochastic Sparse Subspace Clustering

Ying Chen[1], Chun-Guang Li[1], and Chong You[2]
[1] SICE, Beijing University of Posts and Telecommunications
[2] EECS, University of California, Berkeley

Abstract

State-of-the-art subspace clustering methods are based on self-expressive model, which represents each data point as a linear combination of other data points. By enforcing such representation to be sparse, sparse subspace clustering is guaranteed to produce a subspace-preserving data affinity where two points are connected only if they are from the same subspace. On the other hand, however, data points from the same subspace may not be well-connected, leading to the issue of over-segmentation. We introduce dropout to address the issue of over-segmentation, which is based on randomly dropping out data points in self-expressive model. In particular, we show that dropout is equivalent to adding a squared ℓ_2 norm regularization on the representation coefficients, therefore induces denser solutions. Then, we reformulate the optimization problem as a consensus problem over a set of small-scale subproblems. This leads to a scalable and flexible sparse subspace clustering approach, termed Stochastic Sparse Subspace Clustering, which can effectively handle large scale datasets. Extensive experiments on synthetic data and real world datasets validate the efficiency and effectiveness of our proposal.

1. Introduction

In many real world applications, high-dimensional data can be well approximated by a union of low-dimensional subspaces where each subspace corresponds to a class or a category. The problem of segmenting a set of data points according to the subspaces they belong to, known as *subspace clustering* [46, 48], has found many important applications such as motion segmentation [9, 8], image clustering [28], hybrid system identification [45, 3], genes expression profiles clustering [29] and so on.

Prior work. A traditional method for subspace clustering is k-subspaces, which is based on parameterizing a set of basis to the subspaces and finding a segmentation that minimizes the distance of the data points to its corresponding subspaces [5, 2]. The k-subspaces method requires an accurate estimation of the dimension of the underlying subspaces which is not available in many applications. In addition, the associated optimization problem is nonconvex, for which a good initialization is important for finding the optimal solution [23, 16]. Due to the limitations of the k-subspaces methods, modern subspace clustering resorts to *spectral clustering* which recovers the segmentation of data from a proper data affinity graph that captures whether two points are from the same subspace or not. A plethora of early methods for constructing the affinity graph are based on fitting and comparing local subspaces [54, 63]. Such methods require dense samples on the subspaces and cannot handle cases where the subspaces are intersecting.

In the past few years, *self-expressive model* [11, 12] has emerged as a powerful tool for computing affinity graph in subspace clustering and has spurred substantial developments and applications. Given a data matrix $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ whose columns are drawn from a union of subspaces, self-expressive model states that each data point $\boldsymbol{x}_j \in \mathbb{R}^D$ can be expressed as a linear combination of other data points, i.e.,

$$\boldsymbol{x}_j = X\boldsymbol{c}_j + \boldsymbol{e}_j, \quad c_{jj} = 0, \tag{1}$$

where $\boldsymbol{c}_j \in \mathbb{R}^N$ is a coefficient vector and $\boldsymbol{e}_j$ is an error term. While the linear equation in (1) may have many feasible solutions, there exists at least one $\boldsymbol{c}_j$ that is *subspace-preserving*—that is, $\boldsymbol{c}_{ij} \neq 0$ only if points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same subspace [39, 61, 48]. Given subspace-preserving representations $[\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$, the affinity graph is induced by an affinity (weight) matrix whose i, j-th entry is $|c_{ij}| + |c_{ji}|$.

Sparse subspace clustering. Many methods have been proposed for computing subspace-preserving representations by imposing a prior or regularization on the coefficients $\boldsymbol{c}_j$ [12, 28, 10, 24, 40, 60, 55, 58]. Among them, sparse subspace clustering (SSC) [12, 60] that are based on finding the *sparsest* solution to (1) have become extreme popular due to their theoretical guarantees and empirical success. Under mild conditions, SSC is guaranteed to recover subspace-preserving solutions even when data points are corrupted with outliers, noises or missing values and when the subspaces are intersecting or affine

[39, 61, 52, 44, 21, 59].

While subspace-preserving recovery guarantees that no two points from different subspaces are connected in the affinity graph, there is no guarantee that points from the same subspace form a single connected component. Thus, a *connectivity issue* arises that spectral clustering produces an over-segmentation for subspaces with data points that are not well-connected. In particular, an early work [31] shows that the connectivity issue indeed exists in SSC when the dimension of the subspace is greater than 3.

Several works have attempted to address the connectivity issue in SSC. Motivated by the fact that a low-rank regularization on the matrix of coefficients induces dense solutions, a mixture of ℓ_1 and nuclear norm is proposed in [53] to address the connectivity issue. Unfortunately, solving the optimization problem in [53] requires doing singular value decomposition in each iteration of the algorithm, which is computationally prohibitive for large scale data. More recently, in [51] a post-processing step that merges potential over-segmented fragments of a subspace into the same cluster is proposed. While such an approach is conceptually simple and has theoretical guarantees, it only works under the idealized setting where the affinity graph is perfectly subspace-preserving.

Paper Contributions. We exploit *dropout* to address the connectivity issue associated with SSC. *Dropout* is a technique developed for deep learning as an implicit regularization that can effectively alleviate overfitting [41, 50, 49, 4, 13, 7]. In this paper, dropout refers to the operation of dropping out columns of X uniformly at random when computing the self-expressive representation in (1). Such an operation is equivalent to adding an ℓ_2 regularization term on the representation coefficient vector $\boldsymbol{c}_j$, which effectively induces dense solutions. By dropping out columns of the dictionary we solve optimization problems that only involve a (typically very small) part of the original dataset. This is a particularly attractive property when dealing with ultra-large scale datasets that cannot be loaded into memory.

The contributions of the paper are highlighted as follows.

1. We introduce a *dropout* technique into self-expressive model for subspace clustering, and show that it is asymptotically equivalent to a squared ℓ_2 norm regularizer.
2. We propose a *stochastic sparse subspace clustering* model that is based on dropping out columns of the data matrix. The model has flexible scalability and implicit ability to improve the affinity graph connectivity.
3. We reformulate the stochastic sparse subspace clustering model as a consensus optimization problem and develop an efficient consensus algorithm for solving it.
4. We conduct extensive experiments on both synthetic data and real world benchmark data, and demonstrate the state-of-the-art performance of our proposal.

2. Related Work

Self-expressive models in subspace clustering. Existing subspace clustering methods that are based on self-expressive model can be categorized into three groups. a) For the purpose of inducing subspace-preserving solutions, existing methods use different regularizations on $\boldsymbol{c}_j$. This includes the ℓ_1 norm [11], the nuclear norm [25], the ℓ_2 norm [28], the traceLasso norm [27], the ℓ_1 plus nuclear norms [53], the ℓ_1 plus ℓ_2 norms in [58], the ℓ_0 norm in [55] and the weighted ℓ_1 norm in [19, 20]. b) To handle different forms of noise that arise in practical applications, existing methods use different regularizations on $\boldsymbol{e}_j$, e.g., the ℓ_1 and ℓ_2 norms used in [11], the $\ell_{2,1}$ norm used in [25], the mixture of Gaussians in [18], and the weighted error entropy proposed in [22]. c) To perform subspace clustering in an appropriate feature space, self-expressive models are combined with feature learning methods that are based on learning a linear projection [26, 33, 35] or convolution neural networks [14, 64, 62].

Scalable subspace clustering. In recent years, several attempts to address the scalability of subspace clustering have been proposed. For example, in [36], a small subset of data are clustered at first and then the rest of the data are classified based on the learned clusters; in [60, 10], a greedy algorithm [34] is adopted to solve the sparse self-expression model; in [43], a sketching technique is used to speed up SSC; in [56], a divide-and-conquer framework is proposed for extending SSC to large-scale data; in [37], an online dictionary learning based method is proposed to scale up low-rank subspace clustering [47, 25]; in [1], SSC is conducted on a hierarchically clustered multiple subsets of the data and then merged via a multi-layer graphs fusion method; in [57], a greedy exemplar selection approach is proposed to extend SSC to handle class-imbalanced data. While these methods perform subspace clustering on dataset of larger size, there is neither any theoretical guarantee on the quality of the dictionary used in [1, 36, 37] for the purpose of subspace clustering, nor any effort to resolve the connectivity issue of SSC in [10, 60, 56, 43, 1]. As a result, the clustering accuracy in these methods is heavily sacrificed due to using subsampled data or erroneous over-segmentation. Lastly, almost all the subspace clustering methods mentioned above need to load the entire data into memory. If the size of the data is too large, none of these methods still work.

3. Dropout in Self-Expressive Model

We formally introduce the dropout operation to the self-expressive model, and show that it is equivalent to adding an ℓ_2 regularization on the representation vector. In the next section, we use such property of dropout to develop a scalable and flexible subspace clustering model for addressing the graph connectivity issue associated with SSC.

Consider the problem of minimizing the self-expressive residual as follows:

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{x}_j - X\boldsymbol{c}_j\|_2^2, \quad \text{s.t.} \quad c_{jj} = 0. \tag{2}$$

Inspired by the dropout technique used in training neural networks [41, 50, 49, 4, 13, 7], we propose a dropout operation in the self-expressive model in (2). Similar to dropping "hidden neurons" in a neural network, our operation is to discard columns of X uniformly at random.

Specifically, we introduce $0 \leq \delta \leq 1$ as the dropout rate and let $\{\xi_i\}_{i=1}^N$ be N i.i.d. Bernoulli random variables with probability distribution given by

$$\xi_i = \begin{cases} \frac{1}{1-\delta} & \text{with probability } 1-\delta, \\ 0 & \text{with probability } \delta. \end{cases} \tag{3}$$

Then, dropping the columns of X uniformly at random with probability δ in (2) is achieved by multiplication of the N i.i.d. Bernoulli random variables $\{\xi_i\}_i^N$ to the corresponding columns in X, i.e.,

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{x}_j - \sum_i \xi_i c_{ij} \boldsymbol{x}_i\|_2^2 \quad \text{s.t.} \quad c_{jj} = 0. \tag{4}$$

The following theorem gives the asymptotic effect of the dropout in the self-expressive model.

Theorem 1 *Let $\{\xi_i\}_{i=1}^N$ be N i.i.d. Bernoulli random variables with distribution as defined in (3). We have that:*

$$\begin{aligned} &\mathbb{E}_\xi \|\boldsymbol{x}_j - \sum_i \xi_i c_{ij} \boldsymbol{x}_i\|_2^2 \\ &= \|\boldsymbol{x}_j - \sum_i c_{ij} \boldsymbol{x}_i\|_2^2 + \frac{\delta}{1-\delta} \sum_i \|\boldsymbol{x}_i\|_2^2 c_{ij}^2. \end{aligned} \tag{5}$$

By Theorem 1, we can see that the optimization problem

$$\min_{\boldsymbol{c}_j} \mathbb{E}_\xi \|\boldsymbol{x}_j - \sum_i \xi_i c_{ij} \boldsymbol{x}_i\|_2^2 \quad \text{s.t.} \quad c_{jj} = 0, \tag{6}$$

is equivalent to the optimization problem

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{x}_j - \sum_i c_{ij} \boldsymbol{x}_i\|_2^2 + \frac{\delta}{1-\delta} \sum_i \|\boldsymbol{x}_i\|_2^2 c_{ij}^2 \ \text{ s.t. } c_{jj} = 0. \tag{7}$$

In particular, if the columns of X have unit ℓ_2 norm (e.g., by a data preprocessing step), then (7) reduces to

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{x}_j - \sum_i c_{ij} \boldsymbol{x}_i\|_2^2 + \lambda \|\boldsymbol{c}_j\|_2^2 \quad \text{s.t.} \quad c_{jj} = 0, \tag{8}$$

where $\lambda = \frac{\delta}{1-\delta}$. This is precisely the formulation of the subspace clustering method based on least squares regression [28], and is known to yield dense solutions in general.

In this paper, we aim to develop a scalable and flexible subspace clustering method based on the formulation in (6) which, by means of its equivalency to (8), has an implicit ℓ_2 regularization that induces dense solutions. For practical purpose, we replace the expectation $\mathbb{E}_\xi[\cdot]$ with the *sample mean*, and approach the problem in (6) by solving the following optimization problem

$$\min_{\boldsymbol{c}_j} \frac{1}{T} \sum_{t=1}^T \|\boldsymbol{x}_j - \sum_i \xi_i^{(t)} c_{ij} \boldsymbol{x}_i\|_2^2 \quad \text{s.t.} \quad c_{jj} = 0, \tag{9}$$

where $\xi_i^{(t)}$ is the t-th instance of the Bernoulli random variable drawn independently from the distribution in (3).

4. Stochastic Sparse Subspace Clustering: Formulation and A Consensus Algorithm

As briefly discussed in the introduction, sparse subspace clustering aims to find a self-expressive representation with the sparest coefficient vector. That is, it aims to solve the following optimization problem

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{x}_j - X\boldsymbol{c}_j\|_2^2, \quad \text{s.t.} \ \|\boldsymbol{c}_j\|_0 \leq s, \ c_{jj} = 0, \tag{10}$$

where $\|\cdot\|_0$ is the ℓ_0 pseudo-norm that counts the number of nonzero entries in the vector and s is a tuning parameter that controls the sparsity of the solution. It has been shown in [60] that under mild conditions, the greedy algorithm known as Orthogonal Matching Pursuit (OMP) [34] for solving (10) provably produces a subspace-preserving solution. On the other hand, it is also established in [60] that the number of nonzero entries in a subspace-preserving solution produced by OMP cannot exceed the dimension of the subspace that $\boldsymbol{x}_j$ lies in. This upper bound limits the ability of OMP in producing a dense affinity graph, leading to a high risk of over-segmentation.

We incorporate the dropout technique in the previous section to address the connectivity issue in solving (10) via OMP. Specifically, in Section 4.1 we propose a flexible subspace clustering method that combines SSC with (9), and subsequently rewrite it as a consensus optimization problem. Then, in Section 4.2 we present an efficient alternating minimization algorithm to solve the consensus problem.

4.1. Stochastic Sparse Subspace Clustering

By combining the sample mean of the self-expressive model in (9) and the sparsity constraint in (10), we propose a *stochastic sparse subspace clustering* model as follows:

$$\begin{aligned} &\min_{\boldsymbol{c}_j} \frac{1}{T} \sum_{t=1}^T \|\boldsymbol{x}_j - \sum_i \xi_i^{(t)} c_{ij} \boldsymbol{x}_i\|_2^2 \\ &\text{s.t.} \quad \|\boldsymbol{c}_j\|_0 \leq s, \ c_{jj} = 0, \end{aligned} \tag{11}$$

Algorithm 1 : Damped OMP

Input: Dictionary Ξ, $\mathcal{I}$, $\boldsymbol{x}_j \in I\!R^D$, $\boldsymbol{c}_j$, s, λ and ϵ.

1: Initialize $k = 0$, residual $\boldsymbol{q}_j^{(0)} = \boldsymbol{x}_j$, and $S^{(0)} = \emptyset$.
2: **while** $k < s$ and $\|\boldsymbol{q}_j^{(k)}\|_2 > \epsilon$ **do**
3: Find i^* via (16) and update $S^{(k+1)} \leftarrow S^{(k)} \bigcup \{i^*\}$;
4: Update $\boldsymbol{b}_j^{(k+1)}$ by solving (17);
5: Update $\boldsymbol{q}_j^{(k+1)} \leftarrow \boldsymbol{x}_j - \Xi\, \boldsymbol{b}_j^{(k+1)}$ and $k \leftarrow k+1$;
6: **end while**

Output: $\boldsymbol{b}_j^*$

where s controls the sparsity of the solution.[1] Due to the stochastic nature of the dictionaries used in the T subproblems and the sparsity constraint, we refer (11) to Stochastic Sparse Subspace Clustering.

To understand the essence in solving problem (11), we introduce T auxiliary variables $\{\boldsymbol{b}_j^{(t)}\}_{t=1}^T$ and derive an equivalent formulation as follows:

$$\begin{aligned} \min_{\boldsymbol{c}_j, \{\boldsymbol{b}_j^{(t)}\}_{t=1}^T} & \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{x}_j - \sum_i \xi_i^{(t)} b_{ij}^{(t)} \boldsymbol{x}_i\|_2^2, \\ \text{s.t. } & \boldsymbol{b}_j^{(1)} = \cdots = \boldsymbol{b}_j^{(T)} = \boldsymbol{c}_j, \ \ \|\boldsymbol{b}_j^{(t)}\|_0 \le s, \\ & b_{jj}^{(t)} = 0, \ \ t = 1, \cdots, T. \end{aligned} \tag{12}$$

This is clearly a consensus problem over T blocks. Once the optimal solution $\boldsymbol{c}_j$ is found, we induce the affinity via $a_{ij} = \frac{1}{2}(|c_{ij}| + |c_{ji}|)$ and apply spectral clustering via normalized cut [38] on this affinity matrix.

Remark. In problem (12), the T subproblems can be solved in parallel and each subproblem uses a small dictionary with $(1-\delta)N \ll N$ columns on average. This is appealing especially when the data is too large to fit into the memory.

4.2. Consensus Orthogonal Matching Pursuit

To efficiently solve problem (12), instead of solving problem (12) exactly, we introduce a set of penalty terms and solve the relaxed problem as follows:

$$\begin{aligned} \min_{\boldsymbol{c}_j, \{\boldsymbol{b}_j^{(t)}\}} & \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{x}_j - \sum_i \xi_i^{(t)} b_{ij}^{(t)} \boldsymbol{x}_i\|_2^2 + \lambda \|\boldsymbol{b}_j^{(t)} - \boldsymbol{c}_j\|_2^2 \\ \text{s.t. } & \|\boldsymbol{b}_j^{(t)}\|_0 \le s, \ \ b_{jj}^{(t)} = 0, \ \ t = 1, \cdots, T, \end{aligned} \tag{13}$$

where $\lambda > 0$ is a penalty parameter. We solve problem (13) by updating $\{\boldsymbol{b}_j^{(t)}\}_{j=1}^T$ and $\boldsymbol{c}_j$ alternately.

1. **When $\boldsymbol{c}_j$ is fixed:** we solve for $\{\boldsymbol{b}_j^{(t)}\}_{j=1}^T$ in parallel from each of the T subproblems as follows

$$\begin{aligned} \min_{\boldsymbol{b}_j^{(t)}} & \|\boldsymbol{x}_j - \sum_i \xi_i^{(t)} b_{ij}^{(t)} \boldsymbol{x}_i\|_2^2 + \lambda \|\boldsymbol{b}_j^{(t)} - \boldsymbol{c}_j\|_2^2, \\ \text{s.t. } & \|\boldsymbol{b}_j^{(t)}\|_0 \le s, \ \ b_{jj}^{(t)} = 0. \end{aligned} \tag{14}$$

[1] Due to the implicit squared ℓ_2 regularization, the sparsity can be greater than the dimension of the subspace.

Algorithm 2 Consensus OMP for Solving Problem (13)

Input: $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in I\!R^{D \times N}$, $\boldsymbol{x}_j \in I\!R^D$, parameters s, δ, λ, ϵ, T.

1: Sample T subdictionaries $\{\Xi^{(t)}\}_{t=1}^T$ via (3);
2: **while** not converged **do**
3: Given $\boldsymbol{c}_j$, solve T subproblems for $\{\boldsymbol{b}_j^{(t)}\}_{t=1}^T$ in parallel via Algorithm 1;
4: Given $\{\boldsymbol{b}_j^{(t)}\}_{t=1}^T$, update $\boldsymbol{c}_j$ via $\boldsymbol{c}_j \leftarrow \frac{1}{T}\sum_{t=1}^T \boldsymbol{b}_j^{(t)}$;
5: **end while**

Output: $\boldsymbol{c}_j^*$.

Denote the index set for the preserved and dropped columns for the subproblem indexed by t with $\mathcal{I}^{(t)} := \{i : \ \xi_i^{(t)} > 0\}$ and $\mathcal{J}^{(t)} := \{i : \ \xi_i^{(t)} = 0\}$, respectively, and let $\Xi^{(t)}$ be the same as the data matrix X except that columns indexed by J are set to zero vectors. For clarity, we rewrite problem (14) via dictionary $\Xi^{(t)}$ but ignore the superscript t as follows:

$$\begin{aligned} \min_{\boldsymbol{b}_j} & \|\boldsymbol{x}_j - \Xi\, \boldsymbol{b}_j\|_2^2 + \lambda \|\boldsymbol{b}_j - \boldsymbol{c}_j\|_2^2, \\ \text{s.t. } & \|\boldsymbol{b}_j\|_0 \le s, \ \ b_{jj} = 0. \end{aligned} \tag{15}$$

To solve problem (15) efficiently, we develop a greedy algorithm to update $\boldsymbol{b}_j$ from the support within the index set $\mathcal{I}$ of the preserved columns.[2] To be specific, we initialize the support set $S^{(0)}$ as an empty set, the residual $\boldsymbol{q}_j^{(0)} = \boldsymbol{x}_j$, and find the support set $S^{(k+1)}$ of the solution $\boldsymbol{b}_j$ by a greedy search procedure, i.e., incrementing $S^{(k)}$ by adding one index i^* at each iteration via

$$i^* = \arg \max_{i \in \mathcal{I} \setminus S^{(k)}} \psi_i(\boldsymbol{q}_j^{(k)}, \boldsymbol{c}_j), \tag{16}$$

where $\psi_i(\boldsymbol{q}_j^{(k)}, \boldsymbol{c}_j) = (\boldsymbol{x}_i^\top \boldsymbol{q}_j^{(k)})^2 + 2\lambda \boldsymbol{x}_i^\top \boldsymbol{q}_j^{(k)} c_{ij} - \lambda c_{ij}^2$, update $S^{(k+1)} = S^{(k)} \cup \{i^*\}$ to solve problem

$$\begin{aligned} \min_{\boldsymbol{b}_j} & \|\boldsymbol{x}_j - \Xi\, \boldsymbol{b}_j\|_2^2 + \lambda \|\boldsymbol{b}_j - \boldsymbol{c}_j\|_2^2 \\ \text{s.t. } & \text{supp}(\boldsymbol{b}_j) \subseteq S^{(k+1)} \end{aligned} \tag{17}$$

with a closed-form solution, and then compute the residual $\boldsymbol{q}_j^{(k+1)} = \boldsymbol{x}_j - \Xi\, \boldsymbol{b}_j^{(k+1)}$.

We summarize the steps for solving problem (15) in Algorithm 1, termed as Damped Orthogonal Matching Pursuit (Damped OMP).

2. **When $\{\boldsymbol{b}_j^{(t)}\}_{j=1}^T$ are fixed:** we solve $\boldsymbol{c}_j$ from problem

$$\min_{\boldsymbol{c}_j} \frac{\lambda}{T}\sum_{t=1}^{T} \|\boldsymbol{b}_j^{(t)} - \boldsymbol{c}_j\|_2^2, \tag{18}$$

which has a closed-form solution $\boldsymbol{c}_j = \frac{1}{T}\sum_{t=1}^T \boldsymbol{b}_j^{(t)}$.

[2] The reason to update $\boldsymbol{b}_j$ only from the support in $\mathcal{I}$ is to enlarge the support of the consensus solution $\boldsymbol{c}_j$ while keeping the efficiency. This is equivalent to adopt an enlarged sparsity parameter $s' > s$ in (11) or (12).

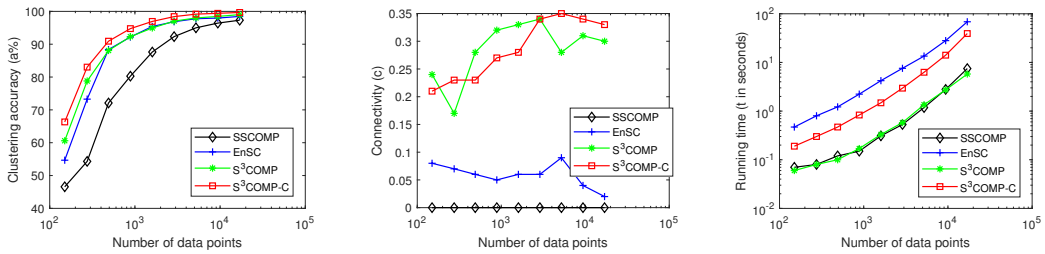


Figure 1. Performance comparison of S^{3}COMP-C, S^{3}COMP, EnSC and SSCOMP on synthetic data.

Algorithm 3 : S^{3}COMP-C

Input: $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in I\!R^{D\times N}$, $\boldsymbol{x}_j \in I\!R^D$, parameters s, δ, λ, ϵ and T.

1: Run Algorithm 2;
2: Define affinity via $a_{ij} = \frac{1}{2}(|c_{ij}| + |c_{ji}|)$;
3: Run spectral clustering via normalized cut [38];

Output: Segmentation matrix.

We summarize the alternating minimization algorithm for solving the consensus problem (13) in Algorithm 2, termed Consensus OMP. For clarity, we sort the whole procedure of our proposed subspace clustering approach in Algorithm 3, termed Stochastic Sparse Subspace Clustering via Orthogonal Matching Pursuit with Consensus (S^{3}COMP-C), and we use S^{3}COMP to refer the approach that solves the consensus problem (13) via Algorithm 2 only one outer iteration. Note that the support size of each solution $\boldsymbol{b}_j^{(t)}$ is up to s and thus the support size of the solution $\boldsymbol{c}_j$ obtained via $\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{b}_j^{(t)}$ will be up to sT, leading to improved connectivity of the induced affinity graph.

Convergence and Stopping Criterion. Similar to the convergence analysis in OMP [34], Algorithm 1 converges in at most s steps. For Algorithm 2, we stop it by checking whether the relative changes of $\boldsymbol{c}_j$ in two successive iterations is smaller than a threshold ε or reaching the maximum iterations. Although we cannot prove the convergence of Algorithm 2, experiments on synthetic data and the real world data demonstrate a good convergence. In experiments, we observe that the number of the outer iterations is small, i.e., $T_0 = 3 \sim 5$ on real world datasets.

Complexity Analysis. In Algorithm 2, it solves T size-reduced subproblems via a damped OMP in parallel, and each subproblem requires $N(1-\delta)$ inner products. Thus, the computation complexity in this stage for each subproblem is $\mathcal{O}(DN^2(1-\delta)s)$ in one outer iteration. The affinity matrix of S^{3}COMP and S^{3}COMP-C contains at most sTN non-zero entries; whereas the affinity matrix of SSCOMP contains at most sN nonzero entries. The eigenvalue decomposition of a sparse matrix using ARPACK requires $\mathcal{O}(snTN)$ operations where n is the number of subspaces (i.e., clusters). While the affinity matrix of S^{3}COMP-C and S^{3}COMP may contain more nonzero entries (up to sTN), the affinity matrix is still sparse and thus the time complexity of eigenvalue decomposition in spectral clustering is $O(nsTN)$, which is slightly higher than $O(nsN)$ of SSCOMP. For a data set of large size, we set $(1-\delta) \ll 1$ and solve T size-reduced subproblems in parallel. This endorses S^{3}COMP-C and S^{3}COMP more flexible scalability.

5. Experiments

To evaluate the performance of our proposed approach, we conduct extensive experiments on both synthetic data and real world benchmark datasets.

Methods and Metrics. We select eight state-of-the-art subspace clustering methods as baselines: SCC [8], LSR [28], LRSC [47], and several scalable subspace clustering methods, including SSCOMP [60], EnSC [58], OLRSC [37], SR-SSC [1], and ESC [57]. In experiments, we use the code provided by the authors for computing the self-expression matrix C in which the parameter(s) is tuned to give the best clustering accuracy. For spectral clustering, we apply the normalized cut [38] on the affinity matrix A which is induced via $A = |C| + |C^\top|$, except for SCC, which has its own spectral clustering step. The reported results in all the experiments of this section are averaged over 10 trials. Following [60], we evaluate each algorithm with clustering accuracy[3] (acc:$a\%$), subspace-preserving representation error (sre:$e\%$), connectivity[4] (conn:c), and running time[5] (t).

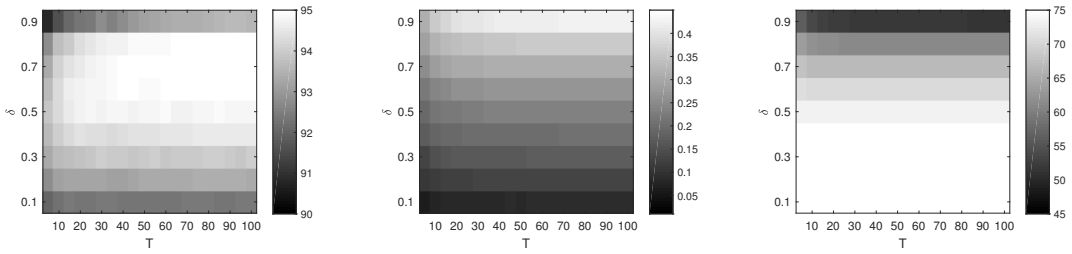
5.1. Experiments on Synthetic Data

Setup. We follow the setting used in [60] to randomly generate $n = 5$ subspaces of dimension $d = 6$ in the ambient space $I\!R^9$. Each subspace contains N_i data points randomly sampled on a unit sphere of $I\!R^9$, in which N_i varies from 30 to 3396. Thus, the total number N of data points varies

[3] It is computed by finding the best alignment between the clustering index and the ground-truth labels under all possible permutations.

[4] Let $\lambda_2^{(i)}$ be the second smallest eigenvalue of the normalized graph Laplacian corresponding to the i-th cluster [30]. The connectivity is computed by $c := \min_i\{\lambda_2^{(i)}\}_{i=1}^n$ for synthetic data. To show the improvement on average for real world data, we compute $\bar{c} := \frac{1}{n}\sum_{i=1}^n \lambda_2^{(i)}$.

[5] The running time of S^{3}COMP in Tables 1 to 5 is based on the maximum running time among T subtasks plus the time of spectral clustering.



(a) Clustering accuracy ($a\%$) (b) Connectivity (c) (c) Subspace-preserving rate ($1-e\%$)

Figure 2. Performance of S^{3}COMP-C as functions of T and δ on synthetic data of $N_i = 320$. The intensity corresponds to the value.

from 150 to 16980. For a fair comparison, we use the same parameter $s = 5$ as in [60]. We set $T = 15$ and select the dropout rate δ in $\{0.1, 0.2, \cdots, 0.9\}$.

We conduct experiments on the synthetic data with different data points per subspace and report the accuracy, connectivity, and subspace-preserving errors. We show each metric as a function of N_i, and present them as curves in Fig. 1. We observe that both S^{3}COMP-C and S^{3}COMP outperform SSCOMP in clustering accuracy and connectivity, especially when the density of data points is lower. It is clear that EnSC, S^{3}COMP and S^{3}COMP-C all improve the connectivity in all cases. The computation time of S^{3}COMP is comparative to (or even lower than) SSCOMP. EnSC yields very competitive clustering accuracy to S^{3}COMP but the time cost is higher than S^{3}COMP-C.

To better understand the effects of parameters δ and T, we conduct experiments with S^{3}COMP-C on synthetic data of $N_i = 320$ under varying $\delta \in \{0.1, \cdots, 0.9\}$ and $T \in \{5, 10, \cdots, 100\}$. The performance of each metric is recorded as a function of δ and T, and displayed as intensity of a gray image in Fig. 2. We observe that the clustering accuracy tends to being stable even using a high dropout rate (e.g., $\delta = 0.85$) whenever T is large than 10. Roughly speaking, higher dropout rate leads to higher connectivity and more efficient algorithm. Thought we also observe that using a higher dropout rate leads to slightly higher subspace-preserving errors[6], it does not necessarily degenerate the clustering accuracy. This is because that the improved connectivity could not only help to avoid over-segmenting the data points in same subspaces but also make the connected data points within the same subspaces have more compact clusters in spectral embedding.

5.2. Experiments on Real World Datasets

In this subsection, we demonstrate the performance of the proposed method on four benchmark datasets, including Extended Yale B (EYaleB) [15], Columbia Object Image Library (COIL100) [32], MNIST [17], and German Traffic Sign Recognition Benchmark (GTSRB) [42].

Dataset Descriptions. Extended Yale B contains 2432 frontal facial images of 38 individuals under 64 different illumination conditions, each of size 192×168. In our experiments, we use the images of all the 38 individuals and resize each image into 48×42 pixels and concatenate the raw pixel in each image as a 2016-dimensional vector.

COIL100 contains 7,200 gray-scale images of 100 different objects. Each object has 72 images taken at pose intervals of 5 degrees. We resize each image to the size 32×32, and concatenate the gray-pixels in each image as a 1024-dimensional vector.

MNIST contains 70,000 grey-scale images of handwritten digits $0-9$. In addition to the whole dataset (denoted MNIST70000), we also prepared two subsets—MNIST4000 and MNIST10000, which are generated by random sampling $N_i = 400$ and $N_i = 1000$ images per category, respectively. For each image, we compute a feature vector of dimension 3,472 using the scattering convolution network [6] and then reduce the dimension to 500 using PCA.

GTSRB contains 43 categories of street sign images with over 50,000 samples in total. We preprocess the dataset as in ESC [57], which results in an imbalanced dataset of 12,390 images in 14 categories. Each image is represented by a 1568-dimensional HOG feature provided with the database. The feature vectors are mean-subtracted and projected to dimension 500 by PCA.

Setup. Note that all feature vectors are normalized to have unit ℓ_2 norm before performing subspace clustering. For a fair comparison, we set $s = 10$ for MNIST and $s = 5$ for Extended Yale B, respectively, as in SSCOMP [60], and set $s = 3$ for GTSRB and COIL100.[7] For the experiments on the real world datasets, we set $T = 15$ and select the dropout rate δ in $\{0.10, 0.20, \cdots, 0.90\}$.

Results. The results on Extended Yale B are listed in Table 1. We can read that S^{3}COMP-C and S^{3}COMP improve the clustering accuracy roughly 10% and 4% over SSCOMP, respectively, and S^{3}COMP-C yields the second best clustering accuracy. The connectivity is improved while

[6] This does not contribute to improve the algebraic connectivity [30]. Thus, the exact relation of the algebraic connectivity with respect to δ is not simply monotonous.

[7] In practice, the parameter s is set to be equal to (or slightly less than) the intrinsic dimension of the data, which could be estimated.

Method	MNIST4000				MNIST10000			
	acc (a%)	sre (e%)	conn ($\bar{c}$)	t (sec.)	acc (a%)	sre (e%)	conn ($\bar{c}$)	t (sec.)
LSR	80.02	78.53	0.6075	14.79	81.75	80.22	0.6389	147.98
LRSC	85.61	79.87	0.6419	4.77	89.60	81.36	0.6646	12.87
SCC	71.30	-	-	70.75	72.20	-	-	218.16
OLRSC	65.32	85.70	**0.8660**	47.4	67.62	86.11	**0.8738**	217.43
ESC	87.22	-	-	27.98	90.76	-	-	59.41
EnSC	85.85	**20.40**	0.1117	35.89	85.94	**16.63**	0.0938	89.21
SSCOMP	91.14	34.26	0.1371	**3.63**	93.80	32.08	0.1212	11.99
SR-SSC	91.70	-	-	39.24	90.05	-	-	79.87
S^{3}COMP	**94.30**	33.15	0.1529	4.70	95.73	30.11	0.1720	**9.14**
S^{3}COMP-C	94.27	33.26	0.1527	12.88	**95.74**	33.15	0.1719	26.50

Table 3. Performance comparison on MNIST where '-' denotes the metric cannot be computed properly.

Method	Extended Yale B			
	acc (a%)	sre (e%)	conn ($\bar{c}$)	t (sec.)
SCC	12.80	-	-	615.69
OLRSC	26.84	95.98	**0.6284**	98.25
LSR	63.99	87.57	0.5067	3.21
LRSC	63.17	88.75	0.4526	7.20
EnSC	61.20	23.14	0.0550	52.98
SR-SSC	62.11	-	-	79.46
SSCOMP	77.59	**20.13**	0.0381	2.54
ESC*	**87.58**	-	-	28.01
S^{3}COMP	81.61	20.18	0.0723	**1.92**
S^{3}COMP-C	87.41	20.28	0.0667	5.05

Table 1. Performance comparison on EYaleB where '-' denotes the metric cannot be computed properly. ESC* uses different way to define affinity from the self-expression coefficients.

Method	COIL100			
	acc (a%)	sre (e%)	conn ($\bar{c}$)	t(sec.)
SCC	55.24	-	-	479.13
LRSC	50.10	96.43	**0.7072**	25.11
LSR	48.22	94.95	0.5246	62.91
SSCOMP	49.88	14.03	0.0060	13.33
ESC	56.90	-	-	56.31
SR-SSC	58.85	-	-	204.38
EnSC	63.94	4.36	0.0163	19.03
S^{3}COMP	71.47	3.35	0.0081	**7.68**
S^{3}COMP-C	**78.89**	**3.15**	0.0077	20.10

Table 2. Performance comparison on COIL100 where '-' denotes the metric cannot be computed properly.

keeping a comparable or even lower subspace-preserving errors and computation cost. While ESC yields the best clustering accuracy, the time cost is much heavier. LSR, LRSC and OLRSC have good connectivity, but the subspace-preserving errors are worse and thus the accuracy is around 60%. While EnSC also has a good connectivity and a low subspace-preserving error, the accuracy and computation time are inferior to S^{3}COMP-C and S^{3}COMP.

In Table 2, we report the results on COIL100. We can read that S^{3}COMP-C and S^{3}COMP yield the leading clustering accuracy and keeping the low subspace-preserving errors. EnSC yields the third best clustering accuracy and subspace-preserving error, and keeps a better connectivity, due to taking a good tradeoff between the ℓ_1 and the ℓ_2 norms. Note that the best three methods S^{3}COMP-C, S^{3}COMP and EnSC all yield very low subspace-preserving error and they share an (implicit or explicit) ℓ_2 norm.

Method	MNIST70000			
	acc (a%)	sre (e%)	conn ($\bar{c}$)	t (sec.)
OLRSC	M	-	-	-
SR-SSC	87.22	-	-	585.31
SSCOMP†	81.59	28.57	0.0830	280.58
ESC	90.87	-	-	596.56
EnSC	93.67	**15.30**	0.0911	932.89
S^{3}COMP†	96.31	30.12	**0.1569**	**218.72**
S^{3}COMP-C†	**96.32**	30.11	**0.1569**	416.84

Table 4. Performance comparison on MNIST where '-' denotes the metric cannot be computed properly, 'M' means that the memory limit of 64G is exceeded. †: The ending eleven eigenvectors associating with the smallest eleven eigenvalues are used in spectral clustering and the details are provided in the supporting material.

The experiments on MNIST are provided in Table 3 and 4. Again, we can observe that S^{3}COMP-C still improves the clustering accuracy around $2 \sim 3\%$ on MNIST4000 and MNIST10000 with improved connectivity than SSCOMP and keeping comparable subspace-preserving errors. On MNIST70000, SSCOMP yields seriously degenerated result than S^{3}COMP-C, S^{3}COMP and EnSC, due to the connectivity issue. While EnSC has the lowest subspace-preserving error, the connectivity and the time cost are not in a good tradeoff. Note that LSR, LRSC, SCC and OLRSC cannot get results because of the memory limit of 64G; whereas S^{3}COMP-C and S^{3}COMP inherit the computation efficiency of SSCOMP.

In Table 5, we show the results on GTSRB. While GTSRB is an imbalanced dataset, surprisingly, we can again observe that the proposed S^{3}COMP and S^{3}COMP-C outperform the listed baseline algorithms and achieve satisfactory results in all four metrics. For EnSC, while it yields the lowest subspace-preserving error, the low connectivity leads to inferior clustering result. Due to the imbalance in

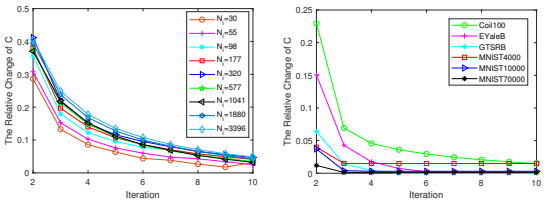
Method	GTSRB			
	acc (a%)	sre (e%)	conn ($\bar{c}$)	t (sec.)
LSR	73.93	82.80	0.6185	290.97
LRSC	87.28	78.97	0.6367	15.85
SCC	70.82	-	-	237.01
OLRSC	82.42	77.15	**0.7606**	291.38
SR-SSC	78.42	-	-	223.34
SSCOMP	82.52	5.42	0.0213	15.43
EnSC	86.05	**0.81**	0.0095	33.46
ESC	90.16	-	-	32.13
S^{3}COMP	95.25	2.40	0.0576	**3.13**
S^{3}COMP-C	**95.54**	2.41	0.0573	7.10

Table 5. Performance comparison on GTSRB where '-' denotes the metric cannot be computed properly.

data distribution, it is hard to find a good tradeoff between the ℓ_1 and ℓ_2 norms.

5.3. More Evaluations

Convergence Behavior. To evaluate the convergence of the proposed S^{3}COMP-C, we show the relative change of the self-expression matrix C in two successive iterations on synthetic data and real world datasets in Fig. 3. We observe that the self-expression matrix becomes stable after a few iterations. This confirms the convergence of Algorithm 2.



(a) Synthetic Data (b) Real World Data

Figure 3. The relative changes of C in successive outer iterations.

Improvements in Connectivity. To better observe the connectivity improvements of the proposed approach, we display the histogram of the second smallest eigenvalues of the normalized graph Laplacian corresponding to each category of GTSRB in Fig. 4. Note that the second minor eigenvalue of a normalized graph Laplacian with respect to each category measures the *algebraic connectivity* [30]. The dramatic improvements in the second minor eigenvalues intuitively indicate significant improvements in connectivity.

Evaluation on Dropout Rate δ. To evaluate the effect of varying the dropout rate δ, we record the performance of S^{3}COMP-C using different dropout rate on synthetic data sets with different number of data points per subspace. Experimental results are presented in Fig. 5. We observe that when the density of the data points increases, the clustering accuracy remains relatively stable when increasing the dropout rate. Thus when the density of data points is higher we can use a larger dropout rate to discard more data points. This confirms that the dropout strategy actually leads to a flexible scalability, while building a desirable tradeoff between the computation efficiency and clustering accuracy.

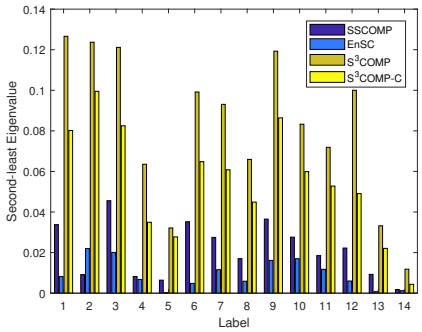
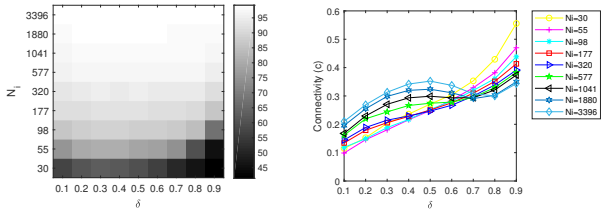


Figure 4. Histograms of second minor eigenvalues of the normalized graph Laplacian for each category of GTSRB.



(a) (b)

Figure 5. Evaluation on effect of dropout rate δ in S^{3}COMP-C on synthetic data of different number of data points per subspace. (a) Clustering accuracy (a%) as a function of δ and N_i. (b) Connectivity c as a function δ.

6. Conclusion

We introduced a dropout strategy in the self-expressive model for subspace clustering. By using Bernoulli random variables, we proved that the dropout in self-expressive model is equivalent to add a squared ℓ_2 norm regularization. Moreover, we proposed a scalable and flexible subspace clustering approach, which is formulated as a consensus optimization problem. We solved the consensus problem by an alternating minimization algorithm which consists of a set of damped orthogonal matching pursuits and an average operation. This leads to a principled and flexible way to improve the connectivity of the induced affinity graph and achieves a desirable tradeoff between the computation efficiency and clustering accuracy. Extensive experiments on synthetic data and real world data have validated the efficiency and the effectiveness of our proposal.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under grant 61876022, and the Open Project Fund from the MOE Key Laboratory of Machine Perception, Peking University.

References

[1] Maryam Abdolali, Nicolas Gillis, and Mohammad Rahmati. Scalable and robust sparse subspace clustering using randomized clustering and multilayer graphs. *Signal Processing*, 163:166–180, 2019. 2, 5

[2] P. Agarwal and N. Mustafa. k-means projective clustering. In *ACM Symposium on Principles of database systems*, 2004. 1

[3] Laurent Bako and René Vidal. Algebraic identification of MIMO SARX models. In *International Workshop on Hybrid Systems: Computation and Control*, pages 43–57. Spinger-Verlag, 2008. 1

[4] P. Baldi and P. Sadowski. Understanding dropout. In *Neural Information Processing Systems*, 2013. 2, 3

[5] P. S. Bradley and O. L. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000. 1

[6] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013. 6

[7] Jacopo Cavazza, Benjamin D Haeffele, Connor Lane, Pietro Morerio, Vittorio Murino, and Rene Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pages 435–444, 2018. 2, 3

[8] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 81(3):317–330, 2009. 1, 5

[9] Joao Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. 1

[10] Eva L. Dyer, Aswin C. Sankaranarayanan, and Richard G. Baraniuk. Greedy feature selection for subspace clustering. *Journal of Machine Learning Research*, 14(1):2487–2517, 2013. 1, 2

[11] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009. 1, 2

[12] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. 1

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016. 2, 3

[14] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, pages 24–33, 2017. 2

[15] Kuang-Chih Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, May 2005. 6

[16] Connor Lane, Benjamin D. Haeffele, and René Vidal. Adaptive online k-subspaces with cooperative re-initialization. In *IEEE International Conference on Computer Vision Workshops*, 2019. 1

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[18] Baohua Li, Ying Zhang, Zhouchen Lin, and Huchuan Lu. Subspace clustering by mixture of gaussian regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2094–2102, 2015. 2

[19] Chun-Guang Li and René Vidal. Structured sparse subspace clustering: A unified optimization framework. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 277–286, 2015. 2

[20] Chun-Guang Li, Chong You, and René Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017. 2

[21] Chun-Guang Li, Chong You, and René Vidal. On geometric analysis of affine sparse subspace clustering. *IEEE Journal on Selected Topics in Signal Processing*, 12(6):1520–1533, 2018. 2

[22] Yuanman Li, Jiantao Zhou, Xianwei Zheng, Jinyu Tian, and Yuan Yan Tang. Robust subspace clustering with independent and piecewise identically distributed noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8720–8729, 2019. 2

[23] John Lipor, David Hong, Dejiao Zhang, and Laura Balzano. Subspace clustering using ensembles of k-subspaces. *arXiv preprint arXiv:1709.04744*, 2017. 1

[24] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013. 1

[25] Guangcan Liu, Zhouchen Lin, and Yingrui Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*, pages 663–670, 2010. 2

[26] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *IEEE International Conference on Computer Vision*, pages 1615–1622, 2011. 2

[27] Can-Yi Lu, Zhouchen Lin, and Shuicheng Yan. Correlation adaptive subspace segmentation by trace lasso. In *IEEE International Conference on Computer Vision*, pages 1345–1352, 2013. 2

[28] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, pages 347–360, 2012. 1, 2, 3, 5

[29] Brian McWilliams and Giovanni Montana. Subspace clustering of high dimensional data: a predictive approach. *Data Mining and Knowledge Discovery*, 28(3):736–772, 2014. 1

[30] Bojan Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991. 5, 6, 8

[31] B. Nasihatkon and R. Hartley. Graph connectivity in sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2137–2144, 2011. 2

[32] Samer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-100). Technical Report CUCS-006-96, Department of Computer Science, Columbia University, February 1996. 6

[33] V. M. Patel, H. V. Nguyen, and R. Vidal. Latent space sparse subspace clustering. In *IEEE International Conference on Computer Vision*, pages 225–232, 2013. 2

[34] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with application to wavelet decomposition. In *Asilomar Conference on Signals, Systems and Computation*, 1993. 2, 3, 5

[35] Chong Peng, Zhao Kang, and Qiang Cheng. Subspace clustering via variance regularized ridge regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2931–2940, 2017. 2

[36] Xi Peng, Lei Zhang, and Zhang Yi. Scalable sparse subspace clustering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2013. 2

[37] Jie Shen, Ping Li, and Huan Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 622–631, 2016. 2, 5

[38] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 4, 5

[39] Mahdi Soltanolkotabi and Emmanuel J. Candès. A geometric analysis of subspace clustering with outliers. *Annals of Statistics*, 40(4):2195–2238, 2012. 1, 2

[40] Mahdi Soltanolkotabi, Ehsan Elhamifar, and Emmanuel J. Candès. Robust subspace clustering. *Annals of Statistics*, 42(2):669–699, 2014. 1

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2, 3

[42] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012. 6

[43] P. A. Traganitis and G. B. Giannakis. Sketched subspace clustering. *IEEE Transactions on Signal Processing*, 66(7):1663–1675, 2018. 2

[44] Manolis C. Tsakiris and René Vidal. Theoretical analysis of sparse subspace clustering with missing entries. In *International Conference on Machine Learning*, pages 4975–4984, 2018. 2

[45] René Vidal. Identification of PWARX hybrid models with unknown and possibly different orders. In *American Control Conference*, pages 547–552, 2004. 1

[46] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(3):52–68, March 2011. 1

[47] René Vidal and Paolo Favaro. Low rank subspace clustering (LRSC). *Pattern Recognition Letters*, 43:47–61, 2014. 2, 5

[48] René Vidal, Yi Ma, and Shankar Sastry. *Generalized Principal Component Analysis*. Springer Verlag, 2016. 1

[49] Stefan Wager, Sida Wang, and Percy S. Liang. Dropout training as adaptive regularization. In *Neural Information Processing Systems*, 2013. 2, 3

[50] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013. 2, 3

[51] Yining Wang, Yu-Xiang Wang, and Aarti Singh. Graph connectivity in noisy sparse subspace clustering. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 538–546, 2016. 2

[52] Yu-Xiang Wang and Huan Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 17(12):1–41, 2016. 2

[53] Yu-Xiang Wang, Huan Xu, and Chenlei Leng. Provable subspace clustering: When LRR meets SSC. In *Neural Information Processing Systems*, 2013. 2

[54] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106, 2006. 1

[55] Yingzhen Yang, Jiashi Feng, Nebojsa Jojic, Jianchao Yang, and Thomas S Huang. ℓ_0-sparse subspace clustering. In *European Conference on Computer Vision*, pages 731–747, 2016. 1, 2

[56] Chong You, Claire Donnat, Daniel P. Robinson, and René Vidal. A divide-and-conquer framework for large-scale subspace clustering. In *Asilomar Conference on Signals, Systems and Computers*, 2016. 2

[57] Chong You, Chi Li, Daniel P. Robinson, and René Vidal. A scalable exemplar-based subspace clustering algorithm for class-imbalanced data. In *European Conference on Computer Vision*, 2018. 2, 5, 6

[58] Chong You, Chun-Guang Li, Daniel Robinson, and René Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3928–3937, 2016. 1, 2, 5

[59] Chong You, Chun-Guang Li, Daniel P. Robinson, and René Vidal. Is an affine constraint needed for affine subspace clustering? In *IEEE International Conference on Computer Vision*, 2019. 2

[60] Chong You, Daniel P. Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3918–3927, 2016. 1, 2, 3, 5, 6

[61] Chong You and René Vidal. Geometric conditions for subspace-sparse recovery. In *International Conference on Machine Learning*, pages 1585–1593, 2015. 1, 2

[62] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. Self-supervised convolutional subspace clustering network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5473–5482, 2019. 2

[63] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision*, 100(3):217–240, 2012. 1

[64] Pan Zhou, Yunqing Hou, and Jiashi Feng. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1596–1604, 2018. 2